Name: DHRUV SINGAL

Table Filling

```
int ySize = y.length + 1;
int xSize = x.length + 1;
int[][] table = new int[ySize][xSize];
// initialize 0th row and column of the table to 2*|row-column|
// because the cost of an empty gene against non empty gene
// is 2* length of nonempty gene
for (int i = 0; i < ySize; i++) {
    table[i][0] = 2 * i;
}
for (int j = 0; j < xSize; j++) {
    table[0][j] = 2 * j;
}
// fill table
/*
* recurrence relation: Ed(i,j)= Ed(i-1,j-1) if match, else
* min((Ed(i,j-1)+2),(Ed(i-1,j)+2))
*/
for (int i = 1; i < ySize; i++) {
    for (int j = 1; j < xSize; j++) {
        if (y[i - 1].equals(x[j - 1])) {
           table[i][j] = table[i - 1][j - 1];
        }
        else {
           table[i][j] = Math.min(table[i - 1][j - 1] + 1, Math.min(table[i
- 1][j] + 2, table[i][j - 1] + 2));
        }
    }
}
```

BackTracing

```
int maxLines = xSize + ySize -2; //-2 because xSize =x.length+1
String[] backTrace = new String[maxLines];
int backTraceIndex = maxLines - 1;
int i = ySize - 1;
int j = xSize - 1;
while (i + j > 0) {
      String temp = "";
      if (table[i][j] == table[i - 1][j - 1] && y[i - 1].equals(x[j - 1]))
      {
      // the characters matched
      // go diagonally up in the table
          temp = x[j - 1] + " " + y[i - 1] + " " + "0";
          backTrace[backTraceIndex] = temp;
          i--;
          j--;
          backTraceIndex--;
      }
      else if (table[i][j] == table[i - 1][j - 1] + 1) {
              // the characters don't match
              // go diagonally up the table
              temp = x[j - 1] + " " + y[i - 1] + " " + "1";
              backTrace[backTraceIndex] = temp;
              i--;
              j--;
              backTraceIndex--;
      }
      else if (table[i][j] == table[i - 1][j] + 2) {
              // x blank
              temp = "- " + y[i - 1] + " " + "2";
              backTrace[backTraceIndex] = temp;
              i--;
              backTraceIndex--;
      }
      else if (table[i][j] == table[i][j - 1] + 2) {
              // y blank
              temp = x[j - 1] + " -" + " " + "2";
              backTrace[backTraceIndex] = temp;
              j--;
              backTraceIndex--;
```

```
      }
}
// print out traceback
for (backTraceIndex=backTraceIndex+1; backTraceIndex < maxLines;
backTraceIndex++) {
    System.out.println(backTrace[backTraceIndex]);
}
```